



[12] 发明专利说明书

[21] ZL 专利号 01144042.2

[45] 授权公告日 2004 年 7 月 28 日

[11] 授权公告号 CN 1159665C

[22] 申请日 2001.12.28 [21] 申请号 01144042.2

[30] 优先权

[32] 2001.1.26 [33] US [31] 09/774,829

[71] 专利权人 国际商业机器公司

地址 美国纽约州

[72] 发明人 史蒂文·V·考夫曼

审查员 刘宇儒

[74] 专利代理机构 北京市柳沈律师事务所

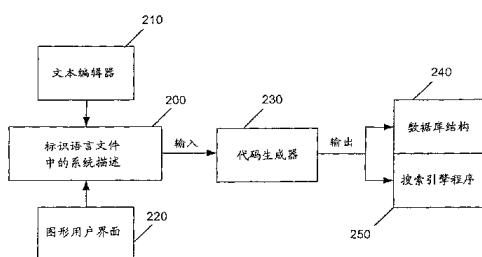
代理人 黄小临 王志森

权利要求书 2 页 说明书 13 页 附图 3 页

[54] 发明名称 创建定制数据库的方法和设备

[57] 摘要

提供了一种用于创建定制数据库的方法。接收数据库的结构的系统描述。根据该系统描述为该定制数据库生成结构。另外，根据该系统描述生成搜索引擎以在该定制数据库中存储和定位数据。



1. 一种在与计算机连接的数据存储装置中创建定制数据库的方法，该方法包括：

5 接收要创建的定制数据库的结构的系统描述；

根据该系统描述生成用于该定制数据库的结构；和

根据该系统描述生成搜索引擎，其中该搜索引擎在该定制数据库中存储和定位数据。

10 2. 如权利要求 1 所述的方法，其特征在于还包括生成用户界面以访问该定制数据库。

3. 如权利要求 1 所述的方法，其特征在于还包括修改该系统描述并生成透明的新的结构和搜索引擎。

4. 如权利要求 1 所述的方法，其特征在于该系统描述定义一个或多个抽象对象到该定制数据库的结构中的物理表示的映射。

15 5. 如权利要求 1 所述的方法，其特征在于包括在该结构中存储数据以形成关系数据库。

6. 如权利要求 1 所述的方法，其特征在于该系统描述包括标识语言文件。

7. 如权利要求 6 所述的方法，其特征在于该标识语言文件包括可扩展标识语言文件。

20 8. 如权利要求 7 所述的方法，其特征在于该可扩展标识语言文件使用文本编辑器创建。

9. 如权利要求 7 所述的方法，其特征在于该可扩展标识语言文件使用图形用户界面创建。

25 10. 如权利要求 1 所述的方法，其特征在于该搜索引擎包括文本搜索引擎。

11. 如权利要求 1 所述的方法，其特征在于该搜索引擎包括高级语言。

12. 如权利要求 11 所述的方法，其特征在于该高级语言包括 Java。

13. 一种用于创建定制数据库的设备，包括：

30 与数据存储装置相连的计算机，其中该数据存储装置用于存储包括所述定制数据库的数据；和

所述计算机包括用于接收要创建的定制数据库的结构的系统描述的装

置；用于根据该系统描述为该定制数据库生成结构的装置；和用于根据该系统描述生成搜索引擎的装置，其中该搜索引擎在该定制数据库中存储和定位数据。

5 14. 如权利要求 13 所述的设备，其特征在于还包括生成用户界面以访问该定制数据库的装置。

15. 如权利要求 13 所述的设备，其特征在于还包括修改该系统描述并生成透明的新的结构和搜索引擎的装置。

16. 如权利要求 13 所述的设备，其特征在于该系统描述定义一个或多个抽象对象到该定制数据库的结构中的物理表示的映射。

10 17. 如权利要求 13 所述的设备，其特征在于还包括在该结构中存储数据以形成关系数据库的装置。

18. 如权利要求 13 所述的设备，其特征在于该系统描述包括标识语言文件。

15 19. 如权利要求 18 所述的设备，其特征在于该标识语言文件包括可扩展标识语言文件。

20. 如权利要求 19 所述的设备，其特征在于该可扩展标识语言文件使用文本编辑器创建。

21. 如权利要求 19 所述的设备，其特征在于该可扩展标识语言文件使用图形用户界面创建。

20 22. 如权利要求 13 所述的设备，其特征在于该搜索引擎包括文本搜索引擎。

23. 如权利要求 13 所述的设备，其特征在于该搜索引擎包括高级语言。

24. 如权利要求 23 所述的设备，其特征在于该高级语言包括 Java。

创建定制数据库的方法和设备

技术领域

5 本发明一般地涉及计算机实现的数据库管理系统，具体地说，涉及使用描述数字图书馆中的数据的系统描述和生成程序的代码生成器创建一个定制数字图书馆，所述该程序根据系统配置对该数字图书馆中的数据进行存储和定位。

10

背景技术

近半个世纪的时间，计算机在商业上用来主要以编码数据的形式管理诸如数字和文本的信息。可是，商业数据仅代表世界信息的一小部分。随着存储、通信和信息处理技术的发展，随着它们成本的降低，数字化其它各种类型的数据、存储大量的数据及能够以按需方式将数据分送给在商业和家庭位置的用户变得更加切实可行。
15

在最近 10 年来已经出现了数字化图像、音频和视频的新的数字化技术，产生了新型的数字多媒体信息。这些多媒体对象与计算机在过去管理的商业数据有很大的不同，并常常需要具有新能力的更先进的信息管理系统基础设施。这种系统常被称为“数字图书馆 (digital library)”。

20

引进新数字技术可以做的事情远远不止仅仅用电子表现替换物理对象。它能够进行对信息的立即访问；支持快速、准确和强大的检索机制；提供新的“基于经验的”（即虚拟现实）用户界面；并实现保护信息所有者权利的新方法。这些特性不仅使数字图书馆解决方案对合作 IS（国际标准化）组织，而且对信息所有者、出版商和服务提供商更具有吸引力和可接受。

25

一般地，商业数据由商业过程产生（例如机票预定、银行存款和在保险公司的索赔处理）。这些过程中的绝大多数已经由计算机实行了自动化，并生成数字形式的商业数据（文本和数字）。因此，它经常被构造为编码数据。相反，因为多媒体数据是人类创造的结果或现实世界的对象的数字化（X 射线、地球物理的绘图等）的结果，而不是计算机算法，所以不能被完全地预构造
30 （pre-structure）（它的使用不是完全可预期的）。

数字形式的商业数据的平均尺寸相对地较小。银行记录（包括客户姓名、

地址、电话号码、帐号、余额等)最多表示为几百个字符,即几百/几千比特。多媒体信息(图像、音频、视频)的数字化产生大量称为“对象”或“斑点(blobs)”(二进制大型对象(Binary Large Objects))的比特。例如,来自梵蒂冈图书馆(Vatican Library)的羊皮纸(parchments)的数字图像占用了相当于三千万字符(30MB)进行存储。甚至在压缩以后,一个电影的数字化可能占用相当于几十亿字符(3-4GB)进行存储。

多媒体信息典型地被作为大对象存储,数量上不断地增加,因此需要特殊的存储机制。传统的商业计算机系统没有设计为直接存储这样大的对象。例如视频或音乐的媒体流的特定类型的信息可能需要特殊化的存储技术。因为特定的多媒体信息需要被“永久”保留,所以它还需要当新的存储技术出现、旧的存储技术过时时,提供备份和移植到新的存储技术的特殊存储管理功能。

最终,由于性能的原因,多媒体数据经常被放置在具有支持多个分布式对象服务器的系统的用户附近。这常常需要在应用程序、标引和数据之间的逻辑分离以确保与数据位置的任何变化无关。

在数字图书馆(DL)中,多媒体对象可以与相关的标引(indexing)信息链接,因为两者都可以利用数字形式。这个传统目录信息与数字化对象的综合是至关重要的并且是数字图书馆技术的很大的优点之一。不同类型的对象可以被不同地分类为适合于每个对象类型。适当的时候可以使用象图书馆的MARC(机读目录)记录、用于特殊收集的归档的Finding Aids(寻找助手)等的现存标准。

面向对象的方法一般地更适合于这种复杂数据管理。术语“面向对象”是指使用模型抽象或真实对象的“分类”和“对象”的软件设计方法。“对象”是面向对象编程的主构件块,是既具有数据又具有功能性(即“方法”)的编程单元。“分类”定义了具体类型对象的实现、该对象使用的变量和方法、和该对象属于的父分类。

当要建造面向客户的数据库时,一般需要与关于数据库内容的客户会晤。然后,硬编码对应于客户需求的对象以建造指定的数据库。换句话说,为每一种客户指定的数据库对象生成编程代码。通过硬编码对象,根据对象的类型将对象存储在预定义的模式(scheme)中。

这种方法有很多明显的缺点。首先,硬编码单独的数据库需要相当的时

间和资金。例如，当要添加新对象到图书馆中时，必须写入新的代码以代表这些对象。类似地，当修改或删除对象时，必须修改数据库并对数据库重新作标引。这个过程导致了用于生成定制数据库的非常低效率的技术。当创建带有大的对象和/或大量对象的大型的、客户指定的数据库时，这些缺点更加突出。
5 因此，硬编码系统的传统手工的方法对于要求在数字图书馆中进行大量对象修改和添加的数据库是效率低下和不理想的。

另外，其它利用基于静态或动态代码生成器的代码生成器的常规系统也有缺点。例如，用基于嵌入静态 SQL（结构化查询语言）的代码生成器，仍必须对每个单独对象、每个添加的对象和每个改变的对象写入代码。另外，
10 对于每个新型数据库请求，必须创建分离的编程语言功能。另外，常规动态嵌入 SQL 代码生成器的工作性能不如静态嵌入 SQL 代码生成器，并一般地针对小型数据库，而不是较大型数据库或定制数据库。

因此，存在对用于使用搜索引擎可以创建和管理定制数据库的系统的技术的需求，该搜索引擎能够有效的运行存储和定位数据库中的数据，而不用
15 生成新的代码、修改已有的代码或对数据库重新作标引以反映数据库内的对象的添加、删除或修改。

发明内容

为了克服上述现有技术的局限，并克服阅读和理解本说明书时将更加明显的其它局限，本发明提供一种在与计算机连接的数据存储装置中创建定制数据库的方法，该方法包括：接收要创建的定制数据库的结构的系统描述；根据该系统描述生成用于该定制数据库的结构；和根据该系统描述生成搜索引擎，其中该搜索引擎在该定制数据库中存储和定位数据。
20

本发明还提供一种用于创建定制数据库的设备，包括：与数据存储装置相连的计算机，其中该数据存储装置用于存储包括所述定制数据库的数据；和所述计算机包括用于接收要创建的定制数据库的结构的系统描述的装置；用于根据该系统描述为该定制数据库生成结构的装置；和用于根据该系统描述生成搜索引擎的装置，其中该搜索引擎在该定制数据库中存储和定位数据。
25

按照本发明，接收了数据库结构的系统描述。基于该系统描述生成定制数据库的结构。生成基于该系统描述的搜索引擎以存储或检索传统数字图书
30

馆或数据库中的数据。

附图说明

下面参照附图，其中相同的标号代表对应的部件：

5 图 1 示意性地说明了本发明的一个实施例的硬件环境，更具体地说，说
明了一种典型的数据库系统；

图 2 示意性地说明了由本发明的一个实施例中的代码生成系统利用的构
件；

10 图 3 是说明本发明的一个实施例中的代码生成系统的操作顺序的
流程图。

具体实施方式

在下面对本发明实施例的描述中，参考了构成本发明一部分的、利用其中
实施本发明的示意性实施例表示的附图。可以理解，在不脱离本发明的范
15 围进行结构改变的情况下，也可以使用其它实施例。

硬件环境

图 1 是用于本发明一个实施例中的硬件环境。本发明的实现包括：计算
机 100，其一般地包括处理器 102、随机存取存储器 (RAM) 104 等；可以存
20 储例如数字图书馆 107 的数据存储装置 106（例如硬盘、软盘和/或 CD-ROM
盘驱动器等）；数据通信装置 108（例如调制解调器、网络接口等）；显示装
置 110（例如 CRT（阴极射线管）、LCD（液晶显示器）显示器等）；输入装置
112（例如鼠标指向装置和键盘）。可以想象，与计算机 100 连接的可以是其
它装置，如只读存储器 (ROM)、视频卡、总线接口、打印机等。本领域的熟
25 练技术人员可以理解，上述装置的任何组合，或任何数量的不同装置、外围
设备和其它装置，可以与计算机 100 一起使用。

计算机 100 在操作系统 (OS) 114 的控制下运行。当计算机 100 接通电
源或重启时，操作系统 114 被导入计算机 100 的存储器 104 用于执行。然后
操作系统的 114 又通过计算机 100 控制一个或多个诸如存储和检索系统 118 的
30 计算机程序 116 的执行。本发明是代码生成系统 119，它用通常的方式在这些
计算机程序 116 中实现，这些程序在操作系统 114 的控制下执行并使得计

计算机 100 执行这里所述的期望的功能。

操作系统 114 和计算机程序 116 包括以下指令：当这些指令由计算机 100 读取和执行时，使得计算机 100 执行必要的步骤以实现和/或使用代码生成系统 119。一般地，操作系统 114 和/或计算机程序 116 从诸如存储器 104、数据存储装置 106、和/或数据通信装置 108 之类的装置、载体、或介质有形地体现和/或读取。在操作系统 114 的控制下，计算机程序 116 可以从存储器 104、数据存储装置 106、和/或数据通信装置 108 装入到计算机 100 的存储器 104，用于在实际操作期间使用。
5

因此，本发明可以实现为使用生产软件、固件、硬件或其任何组合的标准编程和/或工程技术的方法、设备、或制造的产品。这里使用的术语“产品”（或者，“计算机程序产品”）意指包括可从任何计算机可读装置、载体或介质获得的计算机程序。当然，本领域的熟练技术人员可以理解，在不脱离本发明的范围的情况下，可以对这种配置做出很多修改。
10

本领域的熟练技术人员可以理解，图 1 所示的环境不应限制本发明。实际上，本领域的熟练技术人员可以理解，在不脱离本发明的范围的情况下，可以使用其它可选择的硬件环境。
15

XML 概要

可扩展标识语言（XML）是增加了用于创建被称为“XML 文件”的迅速被广泛应用的新的规范。XML 文件包括结构化数据。XML 文件在多种商家之间和商家与客户之间共享。随着关系数据库的长期使用，很多商家将他们的数据存储在关系表中。可是现在，很多商家使用 XML 文件存储数据。
20

代码生成系统 119 利用标识文件，该标识文件包括例如可扩展标识语言（XML）文件。XML 是标准通用标识语言（SGML）的一个子集，SGML 用于带有结构化信息的文件。在 XML 1.0 中描述了 XML，并可以在下列网址找到：
25 <http://www.w3.org/TR/REC-xml>。XML 是一组规则或指南，用于使用标签(tags)设计结构化数据的文本格式。附加的详细信息可以在下列网址找到：
<http://www.w3.org/XML/1999/XML-in-10-points>。XML 是迅速增加了用于创建被称为“XML 文件”的迅速被广泛应用的新的规范，该“XML 文件”可以由各种元素或属性（例如标题、数据、作者等）结构化。一旦以这种方式结构化了文件，就可以基于元素或属性值（或内容）执行结构化搜索。
30

XML 描述了数据对象或 XML 文件的分类并部分地描述了处理它们的计算机程序的行为。XML 文件由称为实体的存储单元构成。每个实体包括经语法分析的和未经语法分析的数据。经语法分析的数据由字符构成，其中的一些形成文件中的字符数据。其它经语法分析的数据可以是标识（markup）的形式。标识将文件的存储布局和逻辑结构的描述编码。XML 提供一种将约束强加于存储布局和逻辑结构上的机制。称为 XML 处理器的软件模块被用于读取 XML 文件并提供对它们的内容和结构的访问。

具体地，可扩展标识语言（XML）是标准通用标识语言（SGML）的一个子集。XML 与可扩展样式表（stylesheet）语言变换（XSLT）和可扩展标识语言路径（Xpath）结合工作。XML 还可以与文件对象模型（DOM）或名称空间（namespace）结合工作。

可扩展标识语言（XML）是标准通用标识语言（SGML）的一个子集。在 XML 1.0 中描述了 XML，并可以在下列网址找到：
<http://www.w3.org/TR/REC-xml>。可扩展标识语言（XML）是一组规则或指南，
用于使用标签（tags）设计结构化数据的文本格式。附加的详细信息可以在
下列网址找到：<http://www.w3.org/XML/1999/XML-in-10-points>。对于互用性，
称作词汇表的域特有标签可以使用文件类型定义（Document Type
Definition）来标准化，以便该域中的应用程序理解这些标签的含义。

可扩展样式语言变换器或 XSLT 是用于将 XML 文件变换为其它 XML 文件的
一种语言。XSLT 规范定义了 XSLT 语言的语法和语意。通过属于被称为 XSLT
名称空间的特定 XML 名称空间来区分 XSLT 定义的元素。表述在 XSLT 中的变
换描述了用于将源树变换为结果树的规则。关于 XSLT 的进一步信息可以在下
列网址找到：<http://www.w3.org/TR/xslt>。

XML 路径或 Xpath 对 XML 文件的各部分编址。Xpath 使用在用于通过 XML
文件的分层结构进行导航的如同 URL（通用资源定位符）中的路径符号
(notation) 获得它的名称。关于 XML 路径的进一步的细节可以在这个网址
找到：<http://www.w3.org/TR/xpath>。

文件目标模型（DOM）是用于从编程语言生成 XML 文件的函数调用的标准
集合。附加的细节可以在下列网址找到：
30 <http://www.w3.org/TR/REC-DOM-Level-1/>。

例如，XML 系统规范的示例提供如下：

```
<?xml encoding="US-ASCII"?>
<!--Digital Library schema definitions -->

<!ELEMENT DLSystem (DLIndexClass+, DLRelationship*, DLAttribute+, DLPart*) >

5
    <!--Index class is a list of relationships, attributes, and parts -->
    <!ELEMENT DLIndexClass (desc) ?>
    <!ATTLIST DLIndexClass
        id ID #REQUIRED
10
        relationships IDREFS #IMPLIED
        dlattributes IDREFS #REQUIRED
        parts IDREFS #IMPLIED>

    <!--Relationships connect index classes -->
15
    <!ELEMENT DLRelationship (desc) ?>
    <!ATTLIST DLRelationship
        id ID #REQUIRED
        type (folder | IDPointer | value) #REQUIRED
        referstoclasses IDREFS #IMPLIED
        fromattribute IDREF #IMPLIED
20
        toattributes IDREFS #IMPLIED>

    <!--attributes are data type declarations
        attributes can be in more than one index class -->
25
    <!ELEMENT DLAttribute (desc) ?>
    <!ATTLIST DLAttribute
        id ID #REQUIRED
        type (VarChar | FixedChar | Integer |
              Time | Long |TimeStamp | Date | Decimal) #REQUIRED
30
        length CDATA #REQUIRED>
```

```

<!-- parts are files or bitstreams -->
<!ELEMENT DLPart (desc)?>
<!ATTLIST DLPart
      id ID #REQUIRED
      partnumber CDATA #REQUIRED
      format CDATA #REQUIRED>
<!ELEMENT desc (#PCDATA)>

```

数字图书馆的代码生成器系统

10 本发明的一个实施例提供了一种代码生成系统 119。代码生成系统 119 为定制数据库创建结构并创建一个或多个在定制数据库中存储和定位数据的程序（例如搜索引擎）。这些程序根据包含定制数据库的结构的系统描述的标识文件来存储和定位数据。

15 图 2 是代码生成系统 119 利用的部件的示意图，说明了将要创建的定制数据库结构的系统描述 200 包括标识语言文件。该规范既可以指“系统描述”，也可以指“标识文件”，并且这两个条目都是指包含系统描述 200 的标识文件。在一个实施例中，标识语言文件 200 可以使用文本编辑器 210 构建。在另一实施例中，标识语言文件 200 可以通过图形用户界面（GUI）220 构建。

20 标识语言文件 200 中的系统描述被输入到代码生成器 230。代码生成器 230 读取标识语言文件 200，根据系统描述创建对象，并输出数据库 240 的结构和一个或多个用于存储和检索数据库中的数据的程序 250（例如搜索引擎程序）。在另一实施例中，代码生成系统可以输出形成数字图书馆 107 的数据库结构。在另一实施例中，数字图书馆可以包括单个的数据库。在另一个实施例中，数字图书馆 107 可以包括数据库的集合。另外，代码生成器 230 输出的搜索引擎程序 250 可以定位定制数据库或数字图书馆 107 中的对象或对象数据。

25 在另一实施例中，尽管图 2 示出了数据库结构 240 和搜索引擎 250 的分离的程序，但同样的程序既可以创建数据库结构 240，又可以用作该定制数据库的搜索引擎 250。下面参照图 3 进一步详细说明代码生成系统 119 利用这些部件的方式。

30 图 3 是说明代码生成系统 119 的方法的流程图。在方块 300 中，代码生

成器 230 接收包含将要创建的定制数据库的结构的系统描述 200 的标识文件。定制数据库的系统描述 200 定义了抽象数字资产 (asset) 到数据库中的物理表示的映射。换句话说，系统描述 200 通过各种对象定义了将要创建的定制数据库的结构。例如，系统描述 200 可以包括，一组将存储在数据库中的数据，数据如何存储，数据如何与其它数据链接或关联，和与数据应该如何存储和/或处理相关的任何其它信息。本领域的熟练技术人员应该理解，大量的其它对象结构和关联可以表示在系统描述 200 中。

标识语言文件存储系统描述 200。例如，系统描述 200 可以输入可扩展标识语言文件 (XML) 的标识。在本发明另一实施例中，系统描述 200 可以输入 Java 文件。系统描述 200 可以使用文本编辑器 210 或通过从图形用户界面 (GUI) 220 输入数据而输入标识文件。本领域的熟练技术人员可以理解，系统描述可以使用其它数据入口系统输入。本发明的一个实施例的例子包括带有系统描述 200 的下列 XML 文件：

```
<?xml version=" 1.0" ?>
15   <!-- sample system structure file for a simple video archive-->
     -<DLSystem name="libsrvrx" userid="frnadmin" password="password">
     -<DLIndexClass dname="VideoAsset">

           <!-- Attributes are database columns -->
20       <DLAttribute id="VA1" datatype="VarChar">Video_TapeNum</DLAttribute>
           <DLAttribute id="VA2" datatype="VarChar">Video_ProxyID</DLAttribute>
           <DLAttribute id="VA3" datatype="VarChar">Video_Date</DLAttribute>
           <DLAttribute id="VA4" datatype="VarChar">Video_Title</DLAttribute>
           <DLAttribute id="VA5" datatype="VarChar">Video_Type</DLAttribute>
25       <DLAttribute id="VA6" datatype="VarChar">Video_IconFile</DLAttribute>
           <DLAttribute id="VA7" datatype="VarChar">Video_NumSBSegments</DLAttribute>
           <DLAttribute      id="VA8"      datatype="FixedChar">Video_TCCorrection<
/DLAttribute>
           <DLAttribute  id="VA9"  datatype = "FixedChar"  defaultvalue = "N"  >
30   Video_Digitized</DLAttribute>
           <DLAttribute id="VA10" datatype="FixedChar" defaultvalue="T">Video_Status<
```

```

/DLAttribute>

    <DLAttribute id="VA11" datatype="FixedChar" defaultvalue = "N ">
Video-CheckedOut </DLAttribute>

    <DLAttribute id="VA12" datatype = "VarChar" > Video_COUserName <
5   /DLAttribute>

        <DLAttribute id="VA13" datatype="Date">Video_CODate</DLAttribute>
        <DLAttribute id="VA14" datatype="Date">Video_CIDate</DLAttribute>
        <DLAttribute id="VA15" datatype="VarChar">Video_ShowNum</DLAttribute>
        <DLAttribute id="VA16" datatype="VarChar">Video_Logger1D</DLAttribute>
10      <DLAttribute id="VA17" datatype="FixedChar">Video_LogDate</DLAttribute>
        <DLAttribute id="VA18" datatype="VarChar">Video_Digitizer1D</DLAttribute>
        <DLAttribute id="VA19"  datatype = "FixedChar" > Video_DigitizeDate
</DLAttribute>

        <DLAttribute id="VA20" datatype="FixedChar">Video_Group</DLAttribute>
15      -<!-- Parts are files stored in object servers-->
        -<!-- transcript file goes into part 10-->
        <DLPart id="trspart" format="txt">10</DLPart>
        -<!-- XML image of asset data goes into part 20, and is indexed by TextMiner-->
        <DLPart id="tmpart" format="xml">20</DLPart>
20      -<!-- Storyboard as text file, is backed up in part 30-->
        <DLPart id="sbpart" format="txt">30</DLPart>
        -<!-- Tar file of .jpgs for storyboard, are backed up in part 40 -->
        <DLPart id="tarpart" format="tar">40</DLPart>
        </DLIndexClass>
25      -<DLIndexClass d1name="Icons">
        <DLAttribute id="PI1" datatype="VarChar">Video_Category</DLAttribute>
        <DLAttribute id="PI2" datatype="VarChar">Video_IconFile</DLAttribute>
        </DLIndexClass>
        </DLSystem>

```

30 是 XML 还是 Java 用于存储系统描述 200，这取决于用户的具体需求和能力。例如，Java 是更直接的语言，对用户来说更容易，可是 XML 是更结构化

的语言，由于更精确的编程需要，它要求用户更多的知识和技巧。另外，更多的现行的系统都是用 Java 编码，而不是用 XML，因此用 Java 编码的文件与现行的系统更兼容。可是，由于最近出现 XML 支持器 (supporter) 的冲击，这种情况在将来可能会有所改变。由于 Java 不可移植到其它系统，并且可能必须分析 Java 程序所做的决定以确定如何由 Java 处理该程序，所以 XML 可能优先于 Java。与需要更多用户分析的 Java 相比，用 XML，用户更能确定 XML 文件正在如何被处理。另外，由于 XML 正在得到流行并正在成为一个描述文件结构的广泛可接受的方法，所以工业标准工具可以使用 XML 文件。因此，不管是使用 XML 还是 Java，都依赖于个人用户的熟练和需求和对 XML 的逐渐接受。本领域的熟练技术人员可以理解，除了 XML 和 Java，其它文件格式也可以结合代码生成系统 119 使用。

在另一实施例中，写入标识语言文件的系统描述 200 的数据可以取表的形式。表的行可以表示要创建的每个对象。表的列可以对应于行中所标识的每个对象的属性或关联。因此，列 1 可以表示存储位置，列 2 可以指示数据被链接到不同的表，等等。

接着在方框 310 中，代码生成器 230 根据系统描述 200 创建定制数据库 240 的结构。数据库结构 240 可以包括诸如提供存储和检索数据的方式的表的各种对象。一个单独的系统描述 200 语句可以生成许多对象。这些对象可以形成一个或多个数据库，数据库又可以形成数字图书馆 107。因此，根据标识文件系统描述 200 中的语句或各个语句，可以生成单独的对象，也可以生成多个对象，或可以生成各对象的整个数据库。

与编程大量的代码行相比，根据系统描述 200 的生成对象可以节省大量的时间。如果定制数据库包括很多对象和/或大型对象，这是尤其实际的。另外，如果要修改数据库，可以通过修改系统描述 200 来修改形成定制数据库的对象，而不是在实现变化以后编码新的对象、重新编码旧对象和对定制数据库的内容重新作标引。

另外，如果有必要，手工地构建没有由代码生成器 230 创建的对象。换句话说，如果代码生成器 230 未根据系统描述 200 产生具体的对象，则这些遗留的对象必须被硬编码并被添加到定制数据库结构，以便创建完整的定制数据库或数字图书馆 107。类似地，未用系统描述 200 实现的删除或任何其它对象的改变必须被硬编码。可是，如果代码生成器 230 可以根据系统描述

200 生成全部的对象，则不需要进一步执行编程以手工地构建对象。

另外，由于系统描述 200 描述定制数据库，可能存在多于一个导致同样定制数据库的描述。换句话说，可以修改系统描述 200，而仍然产生相同的数据库结构 240。

5 在方框 320 中，代码生成器 230 还可以生成在定制数据库中存储数据的程序。根据生成的和任何手工构建的对象，数据被存储到定制数据库并从定制数据库中检索。换句话说，数据库结构 240 将对象包括在可以是例如关系数据库的所需的定制数据库中。当对象被存储在定制数据库时，还可以产生定制数据库内容的标引。

10 不同的系统描述 200 可以描述相同的定制数据库，因此，不同的系统描述可以产生相同的数据库结构 240。换句话说，数据库结构对系统描述 200 中的改变可以是透明的，以便用不同的系统描述 200 生成相同的数据库结构 240。

15 在方框 330 中，代码生成器 230 生成数据库中的定位数据的程序，例如搜索引擎 250。当由搜索引擎程序 250 对定制数据库执行搜索询问时，搜索引擎程序可以是指当存储对象时由数据库程序创建的索引并检索由该询问所请求的对象或对象数据。例如，当由不同的系统描述 200 修改对象，或添加新对象或删除对象时，则数据库程序可以存储或删除这些对象并相应地标引它们。搜索引擎程序可以使用用 Java、超文本标识语言（HTML）或其它高级
20 语言执行的搜索引擎对定制数据库执行询问。另外，搜索引擎程序 250 可以是文本搜索引擎。本领域的熟练技术人员应该理解，代码生成器 230 可以生成可用各种语言格式执行的搜索引擎程序 250。

如前关于数据库结构 240 所描述的，多个系统描述 200 可以描述定制数据库，因此，不同的系统描述 200 可以生成相同的搜索引擎 250。换句话说，
25 搜索引擎 250 对系统描述 200 中的变化可以是透明的。

然后，在方框 340 中，如果需要用户界面以使用户能够访问定制数据库结构 240 或相关搜索引擎 250，则代码生成系统生成用户界面以访问定制数据库。

30 结论

在此总结本发明的实施例的描述。下面描述了一些用于实现本发明的另

外的实施例。例如，诸如大型机、微型机、或个人计算机的任何类型的计算机，或诸如分时大型机、局域网、或独立个人计算机的计算机配置，都可以使用本发明。另外将会理解，用于提供访问控制的面向对象的架构可以用来提供对各种多媒体数据存储器的访问控制，并不限于数字图书馆 107。还可以理解，在不脱离本发明的情况下，在上面的示例分类中描述的功能和方法可以根据需要变化和修改。

为了示例和说明的目的，已经提出了对本发明实施例的前述描述。不打算更加详尽或将本发明限制于所公开的精确的形式。根据上述宗旨，各种修改和变化是可能的。本发明的范围不限于本详细的说明书，而由所附的权利要求来限定。

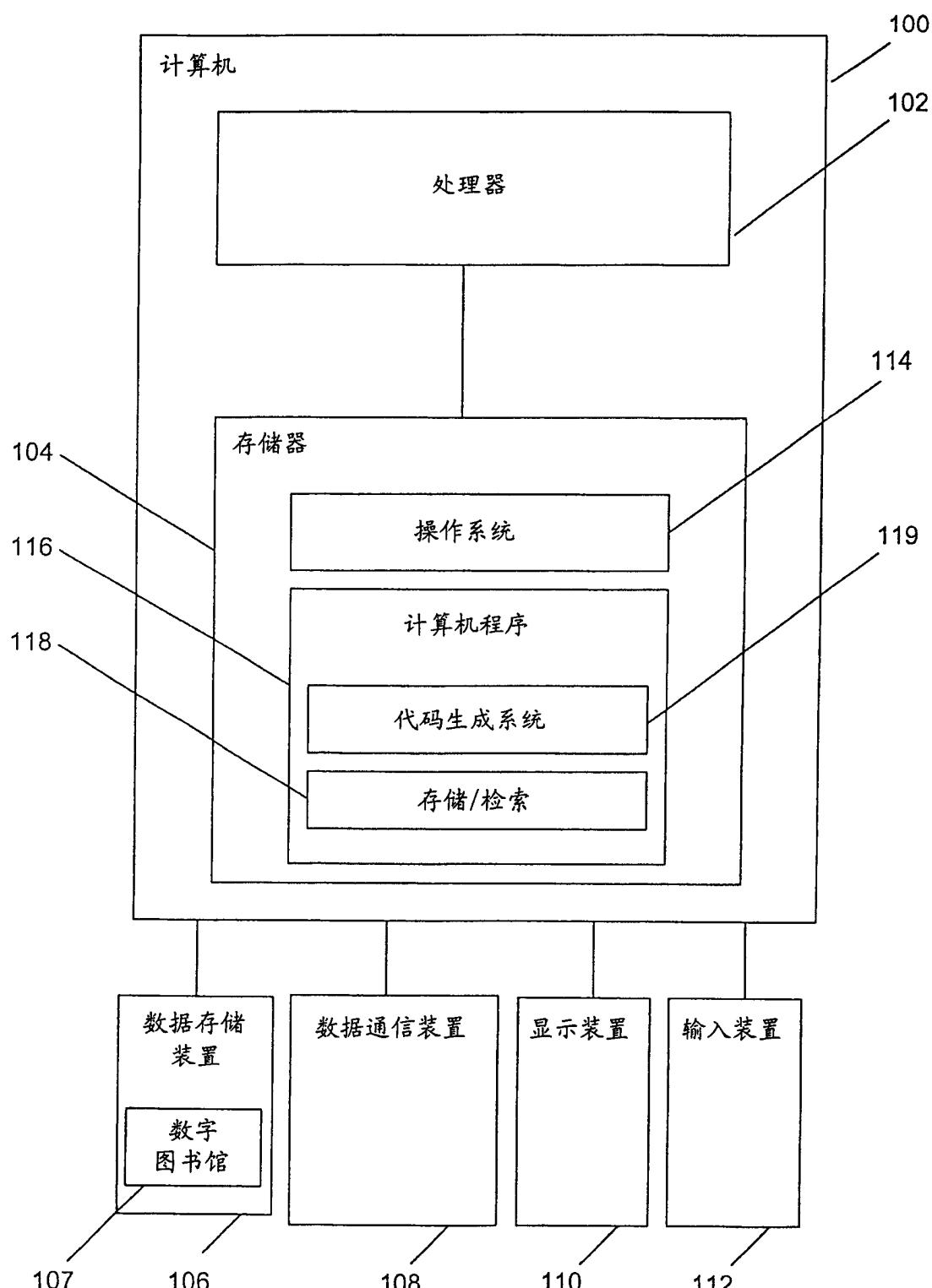


图 1

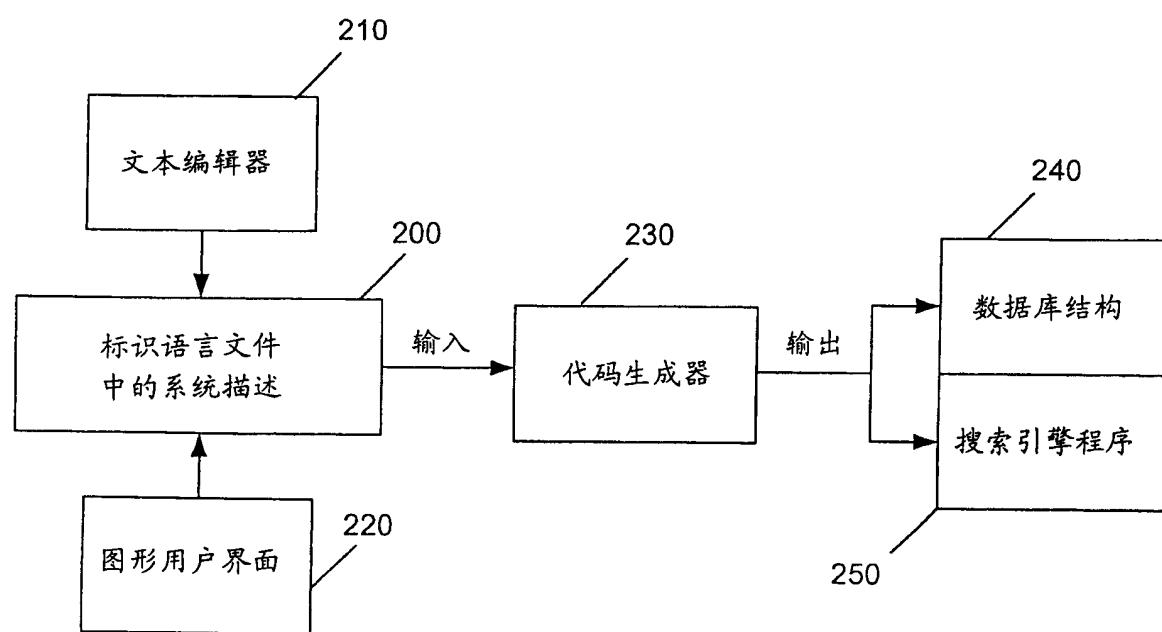


图 2

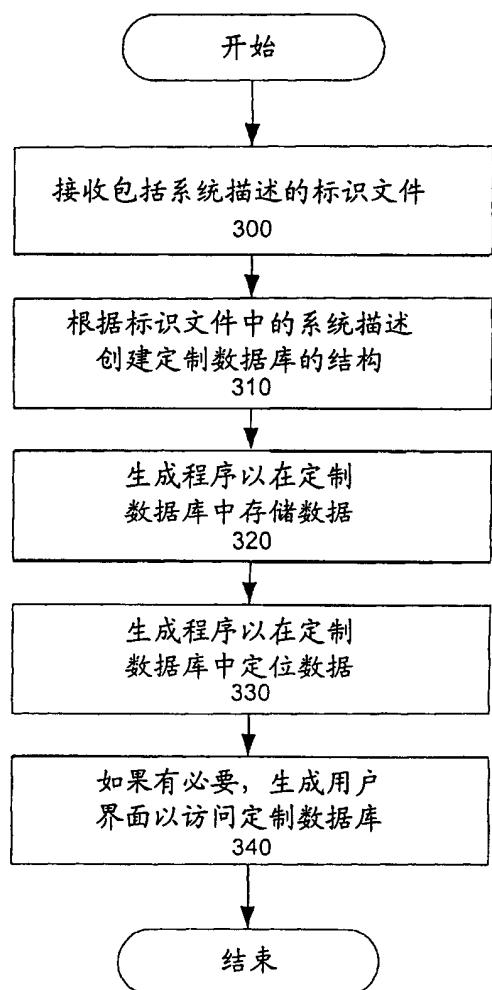


图 3